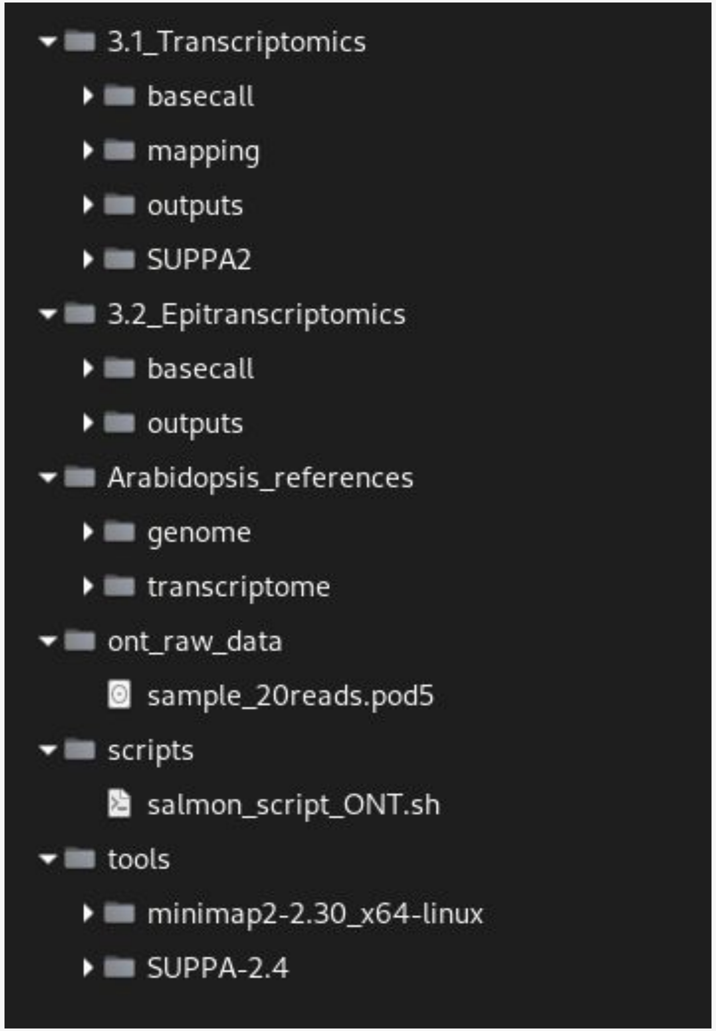
3.1 Transcriptomic-isoforms

Lucas Servi, PhD
Buenos Aires, July 2025

Working directory

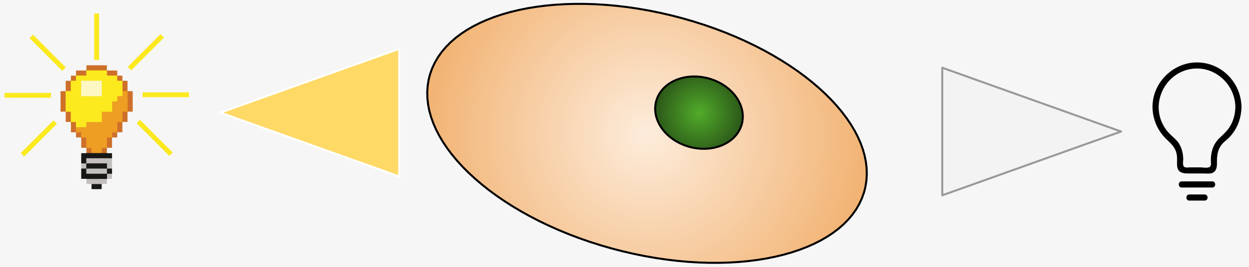
How are our folders structured



```
▾ 3.1_Transcriptomics
    ▸ basecall
    ▸ mapping
    ▸ outputs
    ▸ SUPPA2
▾ 3.2_Epitranscriptomics
    ▸ basecall
    ▸ outputs
▾ Arabidopsis_references
    ▸ genome
    ▸ transcriptome
▾ ont_raw_data
      sample_20reads.pod5
▾ scripts
      salmon_script_ONT.sh
▾ tools
    ▸ minimap2-2.30_x64-linux
    ▸ SUPPA-2.4
```

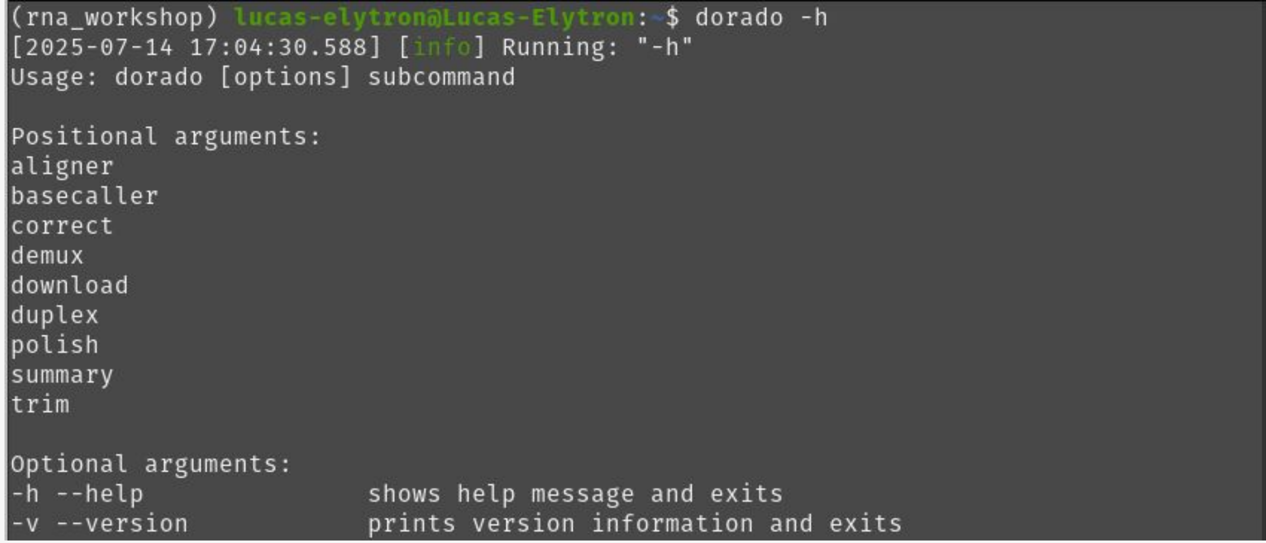
A little bit about the samples
(BARCODES)

1 – 3	Nuclear Fraction Light treatment	4 – 6	Nuclear Fraction Dark treatment
7 – 9	Citoplasm Light treatment	10 – 12	Citoplasm Dark treatment



Dorado Basecaller

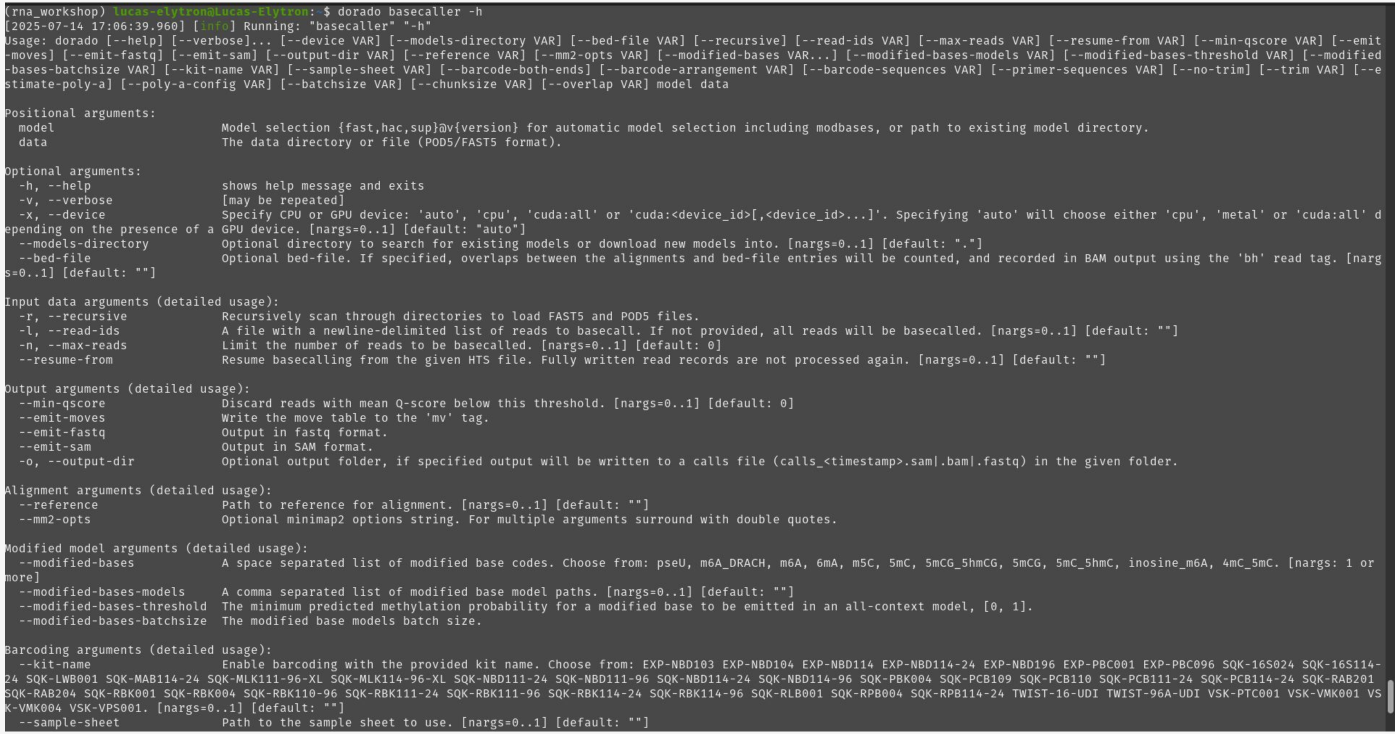
dorado Public



```
(rna_workshop) lucas-elytron@Lucas-Elytron:~$ dorado -h
[2025-07-14 17:04:30.588] [info] Running: "-h"
Usage: dorado [options] subcommand

Positional arguments:
aligner
basecaller
correct
demux
download
duplex
polish
summary
trim

Optional arguments:
-h --help                    shows help message and exits
-v --version                 prints version information and exits
```

https://github.com/nanoporetech/dorado

Dorado Basecaller

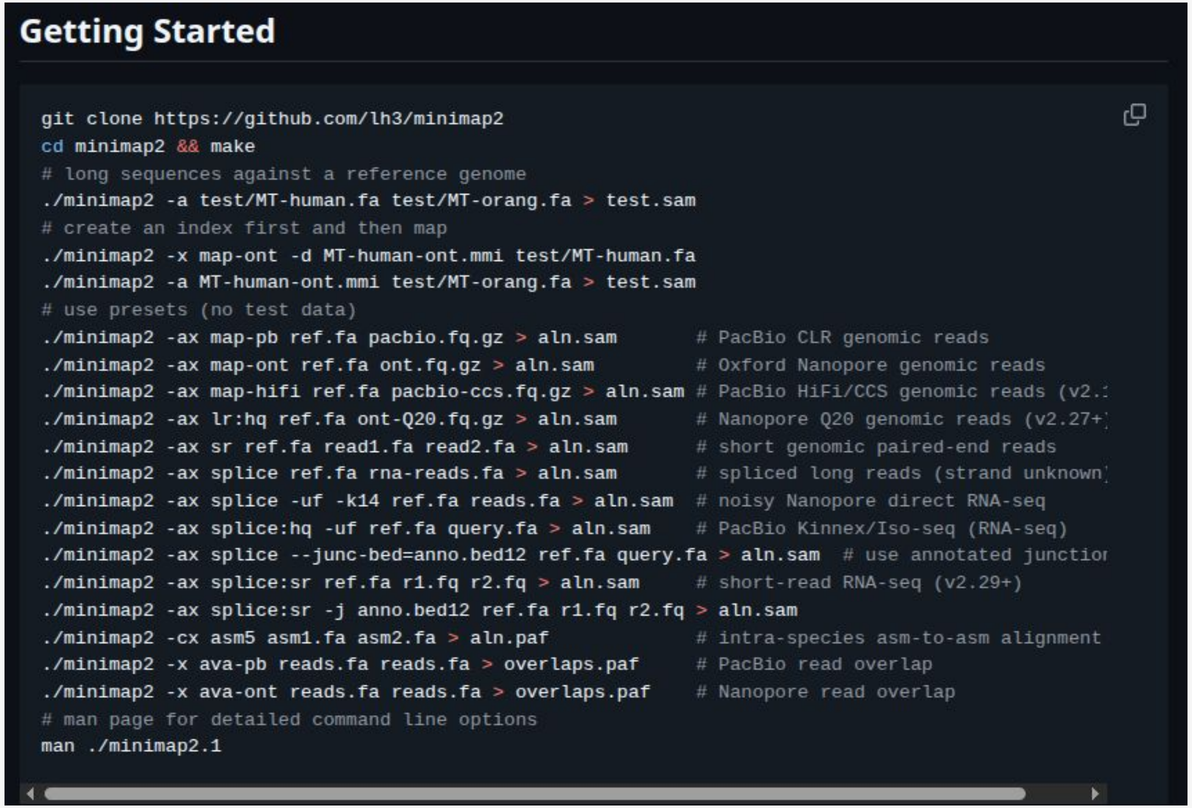


```
(rna_workshop) lucas-elytron@Lucas-Elytron:~$ dorado basecaller -h
[2025-07-14 17:06:39.960] [info] Running: "basecaller" "-h"
Usage: dorado [--help] [--verbose]... [--device VAR] [--models-directory VAR] [--bed-file VAR] [--recursive] [--read-ids VAR] [--max-reads VAR] [--resume-from VAR] [--min-qscore VAR] [--emit
-moves] [--emit-fastq] [--emit-sam] [--output-dir VAR] [--reference VAR] [--mm2-opts VAR] [--modified-bases VAR...] [--modified-bases-models VAR] [--modified-bases-threshold VAR] [--modified
-bases-batchsize VAR] [--kit-name VAR] [--sample-sheet VAR] [--barcode-both-ends] [--barcode-arrangement VAR] [--barcode-sequences VAR] [--primer-sequences VAR] [--no-trim] [--trim VAR] [--e
stimate-poly-a] [--poly-a-config VAR] [--batchsize VAR] [--chunksize VAR] [--overlap VAR] model data

Positional arguments:
  model                         Model selection {fast,hac,sup}@v{version} for automatic model selection including modbases, or path to existing model directory.
  data                          The data directory or file (POD5/FAST5 format).

Optional arguments:
  -h, --help                    shows help message and exits
  -v, --verbose                 [may be repeated]
  -x, --device                  Specify CPU or GPU device: 'auto', 'cpu', 'cuda:all' or 'cuda:<device_id>[,<device_id>...]'. Specifying 'auto' will choose either 'cpu', 'metal' or 'cuda:all' d
epending on the presence of a GPU device. [nargs=0..1] [default: "auto"]
  --models-directory            Optional directory to search for existing models or download new models into. [nargs=0..1] [default: "."]
  --bed-file                    Optional bed-file. If specified, overlaps between the alignments and bed-file entries will be counted, and recorded in BAM output using the 'bh' read tag. [narg
s=0..1] [default: ""]

Input data arguments (detailed usage):
  -r, --recursive               Recursively scan through directories to load FAST5 and POD5 files.
  -l, --read-ids                A file with a newline-delimited list of reads to basecall. If not provided, all reads will be basecalled. [nargs=0..1] [default: ""]
  -n, --max-reads               Limit the number of reads to be basecalled. [nargs=0..1] [default: 0]
  --resume-from                 Resume basecalling from the given HTS file. Fully written read records are not processed again. [nargs=0..1] [default: ""]

Output arguments (detailed usage):
  --min-qscore                  Discard reads with mean Q-score below this threshold. [nargs=0..1] [default: 0]
  --emit-moves                  Write the move table to the 'mv' tag.
  --emit-fastq                  Output in fastq format.
  --emit-sam                    Output in SAM format.
  -o, --output-dir              Optional output folder, if specified output will be written to a calls file (calls_<timestamp>.sam|.bam|.fastq) in the given folder.

Alignment arguments (detailed usage):
  --reference                   Path to reference for alignment. [nargs=0..1] [default: ""]
  --mm2-opts                    Optional minimap2 options string. For multiple arguments surround with double quotes.

Modified model arguments (detailed usage):
  --modified-bases              A space separated list of modified base codes. Choose from: pseU, m6A_DRACH, m6A, 6mA, m5C, 5mC, 5mCG_5hmCG, 5mCG, 5mC_5hmC, inosine_m6A, 4mC_5mC. [nargs: 1 or
more]
  --modified-bases-models       A comma separated list of modified base model paths. [nargs=0..1] [default: ""]
  --modified-bases-threshold    The minimum predicted methylation probability for a modified base to be emitted in an all-context model, [0, 1].
  --modified-bases-batchsize    The modified base models batch size.

Barcoding arguments (detailed usage):
  --kit-name                    Enable barcoding with the provided kit name. Choose from: EXP-NBD103 EXP-NBD104 EXP-NBD114 EXP-NBD114-24 EXP-NBD196 EXP-PBC001 EXP-PBC096 SQK-16S024 SQK-16S114-
24 SQK-LWB001 SQK-MAB114-24 SQK-MLK111-96-XL SQK-MLK114-96-XL SQK-NBD111-24 SQK-NBD111-96 SQK-NBD114-24 SQK-NBD114-96 SQK-PBK004 SQK-PCB109 SQK-PCB110 SQK-PCB111-24 SQK-PCB114-24 SQK-RAB201
SQK-RAB204 SQK-RBK001 SQK-RBK004 SQK-RBK110-96 SQK-RBK111-24 SQK-RBK111-96 SQK-RBK114-24 SQK-RBK114-96 SQK-RLB001 SQK-RPB004 SQK-RPB114-24 TWIST-16-UDI TWIST-96A-UDI VSK-PTC001 VSK-VMK001 VS
K-VMK004 VSK-VPS001. [nargs=0..1] [default: ""]
  --sample-sheet                Path to the sample sheet to use. [nargs=0..1] [default: ""]
```

https://github.com/nanoporetech/dorado

https://github.com/lh3/minimap2

Mapping the reads – Minimap2

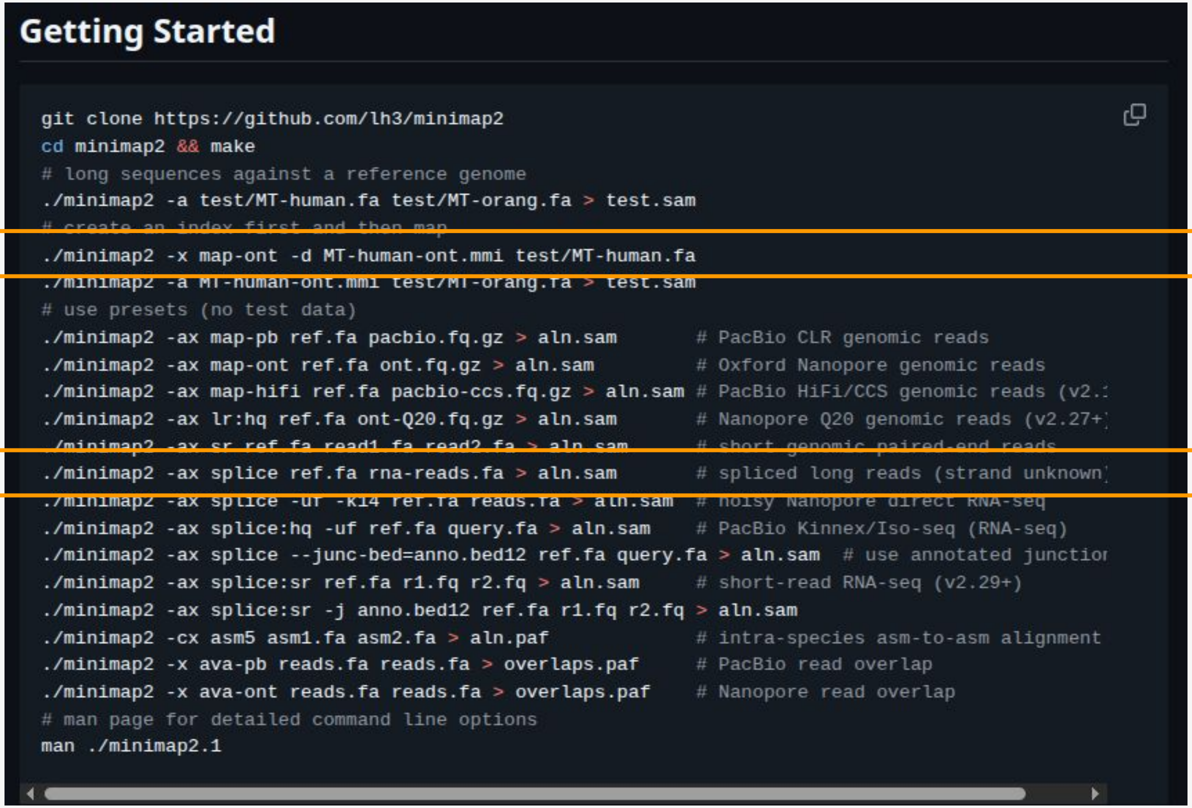


Getting Started

```
git clone https://github.com/lh3/minimap2
cd minimap2 && make
# long sequences against a reference genome
./minimap2 -a test/MT-human.fa test/MT-orang.fa > test.sam
# create an index first and then map
./minimap2 -x map-ont -d MT-human-ont.mmi test/MT-human.fa
./minimap2 -a MT-human-ont.mmi test/MT-orang.fa > test.sam
# use presets (no test data)
./minimap2 -ax map-pb ref.fa pacbio.fq.gz > aln.sam       # PacBio CLR genomic reads
./minimap2 -ax map-ont ref.fa ont.fq.gz > aln.sam         # Oxford Nanopore genomic reads
./minimap2 -ax map-hifi ref.fa pacbio-ccs.fq.gz > aln.sam # PacBio HiFi/CCS genomic reads (v2.
./minimap2 -ax lr:hq ref.fa ont-Q20.fq.gz > aln.sam       # Nanopore Q20 genomic reads (v2.27+
./minimap2 -ax sr ref.fa read1.fa read2.fa > aln.sam      # short genomic paired-end reads
./minimap2 -ax splice ref.fa rna-reads.fa > aln.sam       # spliced long reads (strand unknown
./minimap2 -ax splice -uf -k14 ref.fa reads.fa > aln.sam  # noisy Nanopore direct RNA-seq
./minimap2 -ax splice:hq -uf ref.fa query.fa > aln.sam    # PacBio Kinnex/Iso-seq (RNA-seq)
./minimap2 -ax splice --junc-bed=anno.bed12 ref.fa query.fa > aln.sam  # use annotated junctio
./minimap2 -ax splice:sr ref.fa r1.fq r2.fq > aln.sam     # short-read RNA-seq (v2.29+)
./minimap2 -ax splice:sr -j anno.bed12 ref.fa r1.fq r2.fq > aln.sam
./minimap2 -cx asm5 asm1.fa asm2.fa > aln.paf             # intra-species asm-to-asm alignment
./minimap2 -x ava-pb reads.fa reads.fa > overlaps.paf     # PacBio read overlap
./minimap2 -x ava-ont reads.fa reads.fa > overlaps.paf    # Nanopore read overlap
# man page for detailed command line options
man ./minimap2.1
```

https://github.com/lh3/minimap2

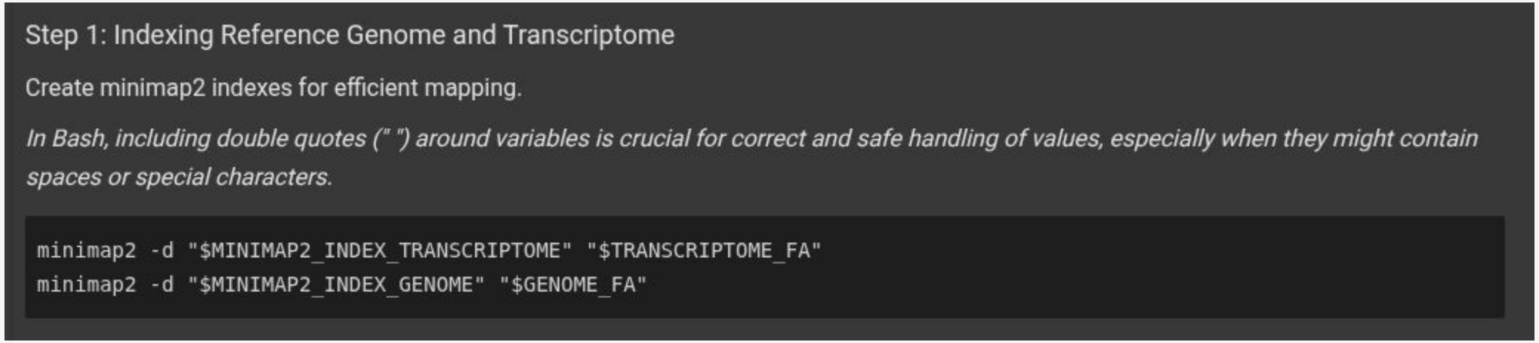
Mapping the reads – Minimap2



Getting Started

```
git clone https://github.com/lh3/minimap2
cd minimap2 && make
# long sequences against a reference genome
./minimap2 -a test/MT-human.fa test/MT-orang.fa > test.sam
# create an index first and then map
./minimap2 -x map-ont -d MT-human-ont.mmi test/MT-human.fa
./minimap2 -a MT-human-ont.mmi test/MT-orang.fa > test.sam
# use presets (no test data)
./minimap2 -ax map-pb ref.fa pacbio.fq.gz > aln.sam       # PacBio CLR genomic reads
./minimap2 -ax map-ont ref.fa ont.fq.gz > aln.sam         # Oxford Nanopore genomic reads
./minimap2 -ax map-hifi ref.fa pacbio-ccs.fq.gz > aln.sam # PacBio HiFi/CCS genomic reads (v2.
./minimap2 -ax lr:hq ref.fa ont-Q20.fq.gz > aln.sam       # Nanopore Q20 genomic reads (v2.27+
./minimap2 -ax sr ref.fa read1.fa read2.fa > aln.sam      # short genomic paired-end reads
./minimap2 -ax splice ref.fa rna-reads.fa > aln.sam       # spliced long reads (strand unknown
./minimap2 -ax splice -uf -k14 ref.fa reads.fa > aln.sam  # noisy Nanopore direct RNA-seq
./minimap2 -ax splice:hq -uf ref.fa query.fa > aln.sam    # PacBio Kinnex/Iso-seq (RNA-seq)
./minimap2 -ax splice --junc-bed=anno.bed12 ref.fa query.fa > aln.sam  # use annotated junctio
./minimap2 -ax splice:sr ref.fa r1.fq r2.fq > aln.sam     # short-read RNA-seq (v2.29+)
./minimap2 -ax splice:sr -j anno.bed12 ref.fa r1.fq r2.fq > aln.sam
./minimap2 -cx asm5 asm1.fa asm2.fa > aln.paf             # intra-species asm-to-asm alignment
./minimap2 -x ava-pb reads.fa reads.fa > overlaps.paf     # PacBio read overlap
./minimap2 -x ava-ont reads.fa reads.fa > overlaps.paf    # Nanopore read overlap
# man page for detailed command line options
man ./minimap2.1
```

What’s the script doing?



Step 1: Indexing Reference Genome and Transcriptome

Create minimap2 indexes for efficient mapping.

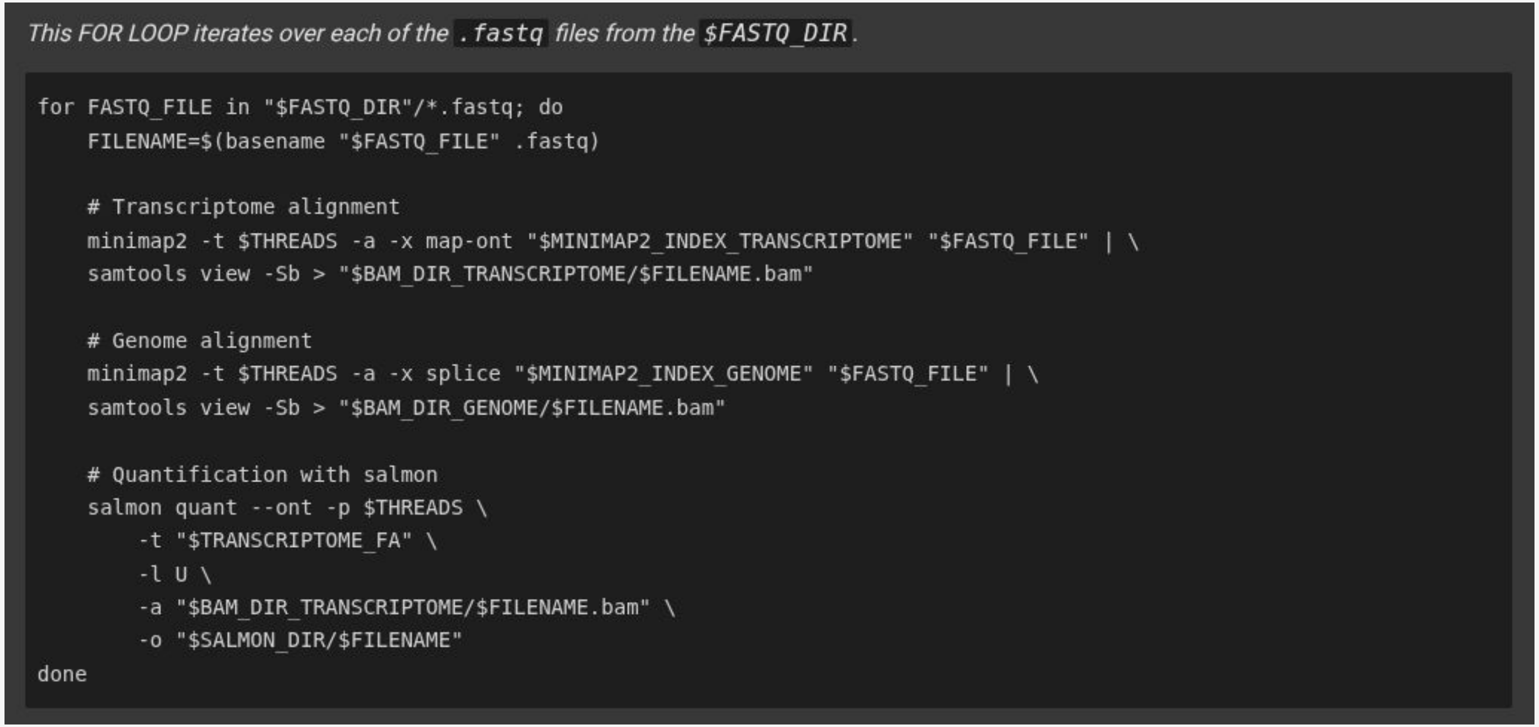
In Bash, including double quotes (" ") around variables is crucial for correct and safe handling of values, especially when they might contain spaces or special characters.

```
minimap2 -d "$MINIMAP2_INDEX_TRANSCRIPTOME" "$TRANSCRIPTOME_FA"
minimap2 -d "$MINIMAP2_INDEX_GENOME" "$GENOME_FA"
```

Each line is creating an index, one for the transcriptome and the other for the genome.

This speeds up future mapping, and it’s key when using the same reference for multiple samples.

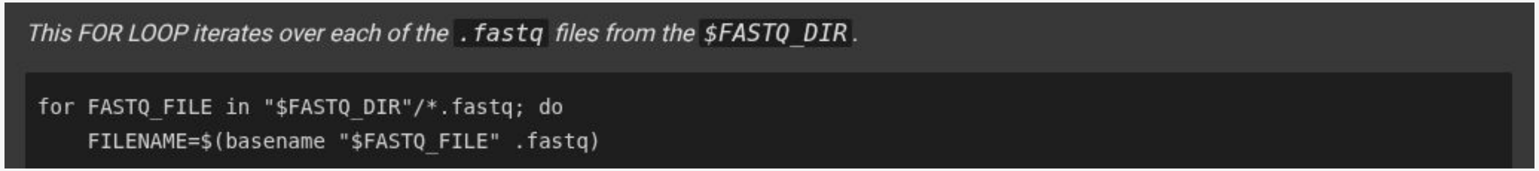
What's the script doing?



This FOR LOOP iterates over each of the `.fastq` files from the `$FASTQ_DIR`.

```
for FASTQ_FILE in "$FASTQ_DIR"/*.fastq; do
    FILENAME=$(basename "$FASTQ_FILE" .fastq)

    # Transcriptome alignment
    minimap2 -t $THREADS -a -x map-ont "$MINIMAP2_INDEX_TRANSCRIPTOME" "$FASTQ_FILE" | \
    samtools view -Sb > "$BAM_DIR_TRANSCRIPTOME/$FILENAME.bam"

    # Genome alignment
    minimap2 -t $THREADS -a -x splice "$MINIMAP2_INDEX_GENOME" "$FASTQ_FILE" | \
    samtools view -Sb > "$BAM_DIR_GENOME/$FILENAME.bam"

    # Quantification with salmon
    salmon quant --ont -p $THREADS \
        -t "$TRANSCRIPTOME_FA" \
        -l U \
        -a "$BAM_DIR_TRANSCRIPTOME/$FILENAME.bam" \
        -o "$SALMON_DIR/$FILENAME"
done
```

What’s the script doing?



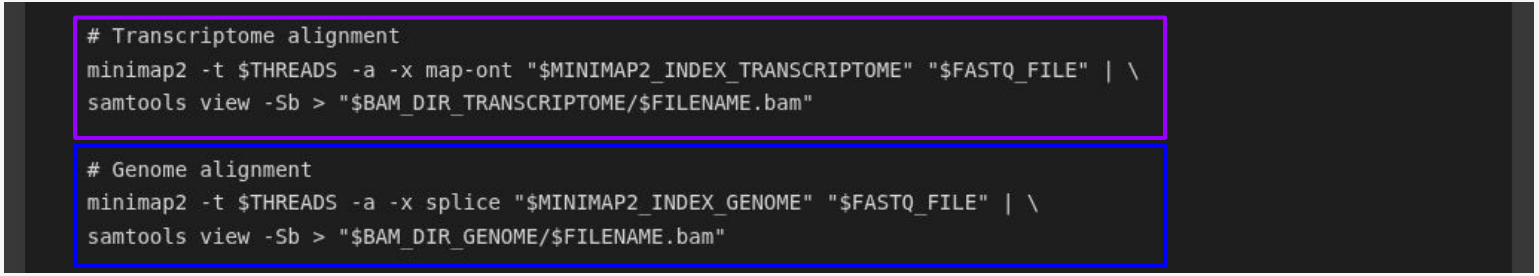
This FOR LOOP iterates over each of the `.fastq` *files from the* `$FASTQ_DIR`*.*

```
for FASTQ_FILE in "$FASTQ_DIR"/*.fastq; do
    FILENAME=$(basename "$FASTQ_FILE" .fastq)
```

Iterates over each file on the directory ending in `.fastq` and assigns the file name to the variable `FASTQ_FILE`.
The “” is a wildcard, meaning “everything”.*

Then, the extension is removed, and the file name is saved to the variable `FILENAME`.

What’s the script doing?



```
# Transcriptome alignment
minimap2 -t $THREADS -a -x map-ont "$MINIMAP2_INDEX_TRANSCRIPTOME" "$FASTQ_FILE" | \
samtools view -Sb > "$BAM_DIR_TRANSCRIPTOME/$FILENAME.bam"

# Genome alignment
minimap2 -t $THREADS -a -x splice "$MINIMAP2_INDEX_GENOME" "$FASTQ_FILE" | \
samtools view -Sb > "$BAM_DIR_GENOME/$FILENAME.bam"
```

Now we are mapping (with minimap2) to the reference transcriptome and to the **reference genome**.

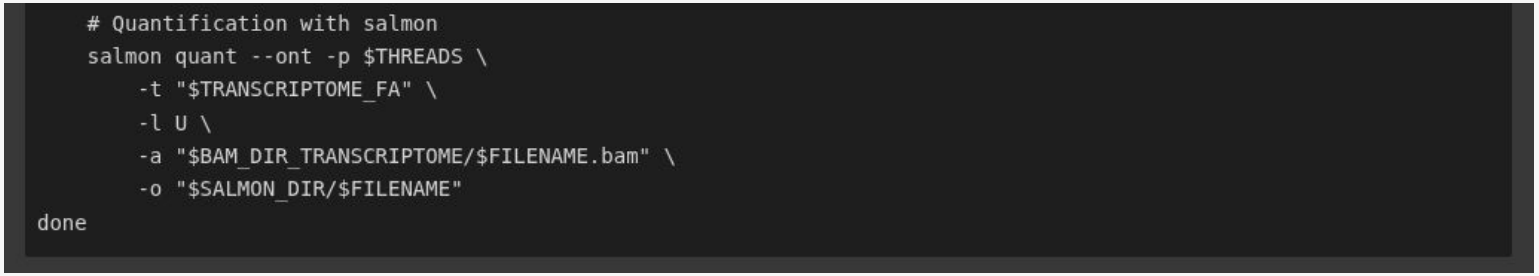
After each mapping, we use `samtools` to *compress* the **.SAM (Sequence Alignment/Map)** to a **.BAM (binary version of the Sequence Alignment/Map)**.

What's the script doing?

When using SALMON to quantify isoforms from long reads, a previous mapping is required (not necessary with illumina reads).

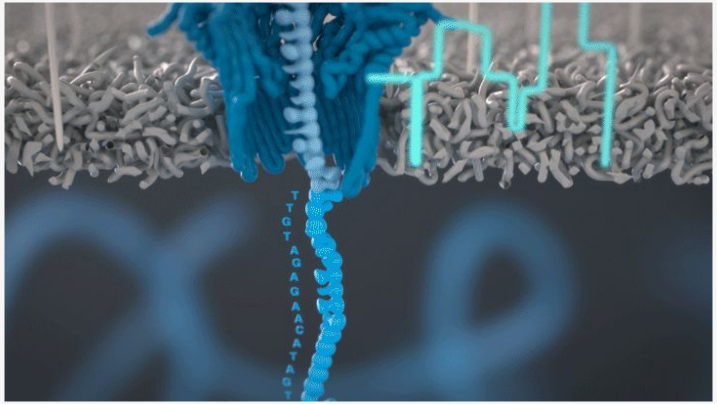
Here we quantify with salmon setting the parameters for `--ont` reads.

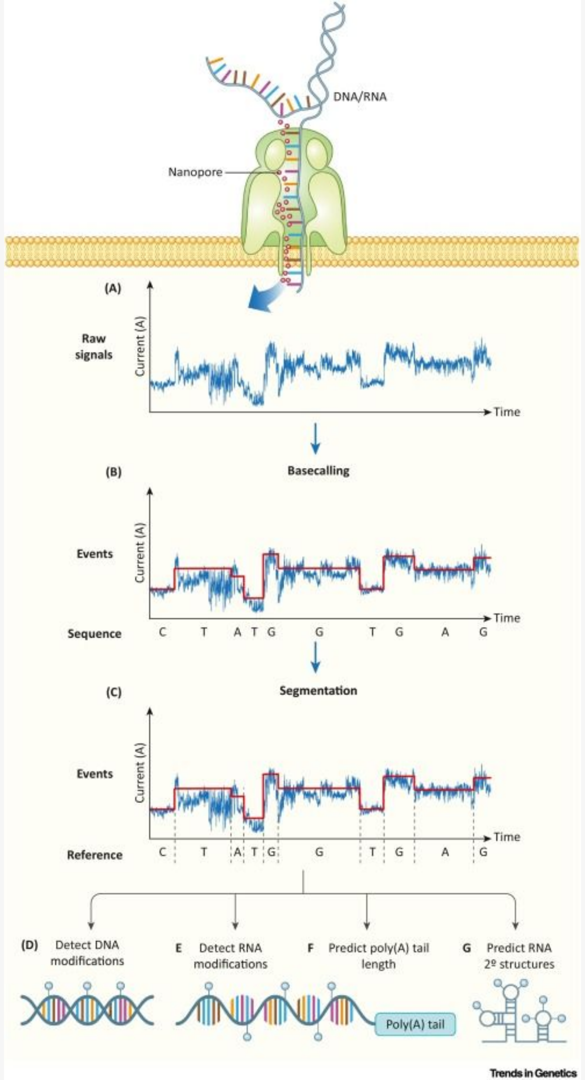
`-l U` refers to unstranded library



```
    # Quantification with salmon
    salmon quant --ont -p $THREADS \
        -t "$TRANSCRIPTOME_FA" \
        -l U \
        -a "$BAM_DIR_TRANSCRIPTOME/$FILENAME.bam" \
        -o "$SALMON_DIR/$FILENAME"
done
```

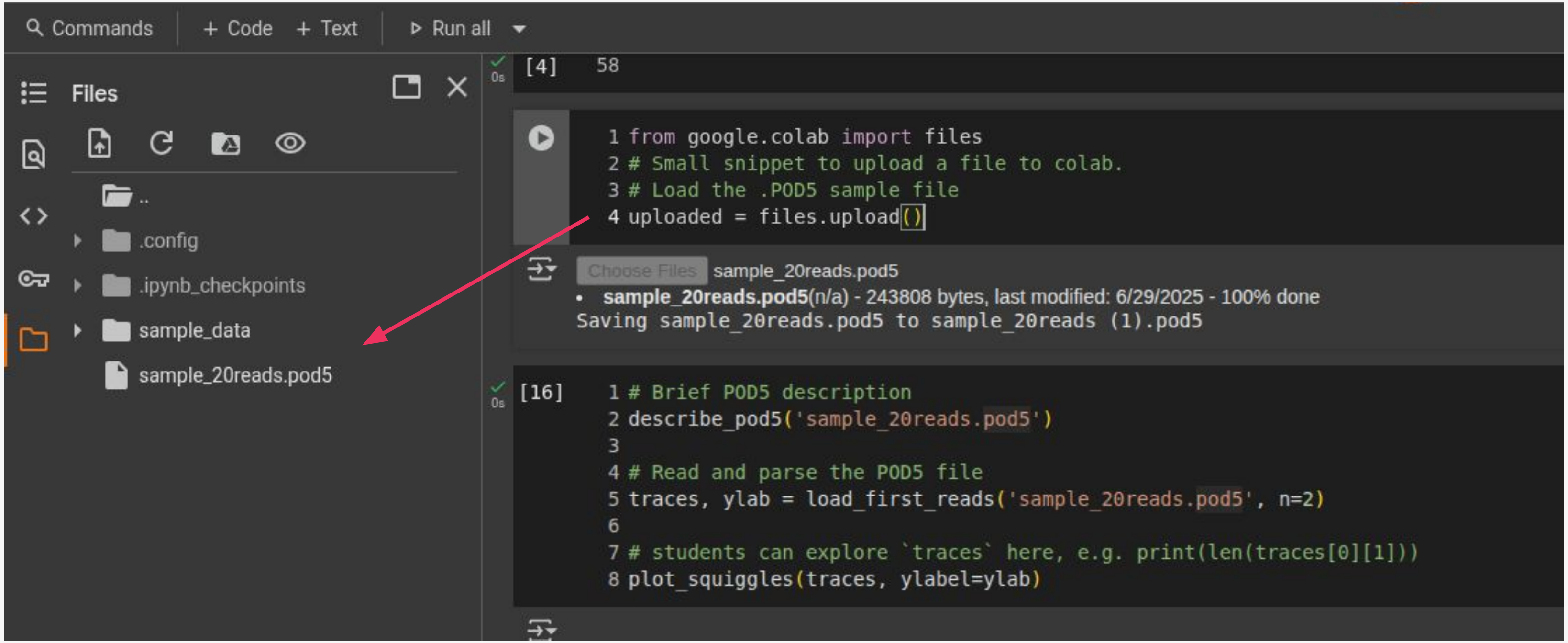
How does Nanopores work?





Comments on the code

Uploading files to work with Colab:



```
[4]   58
```

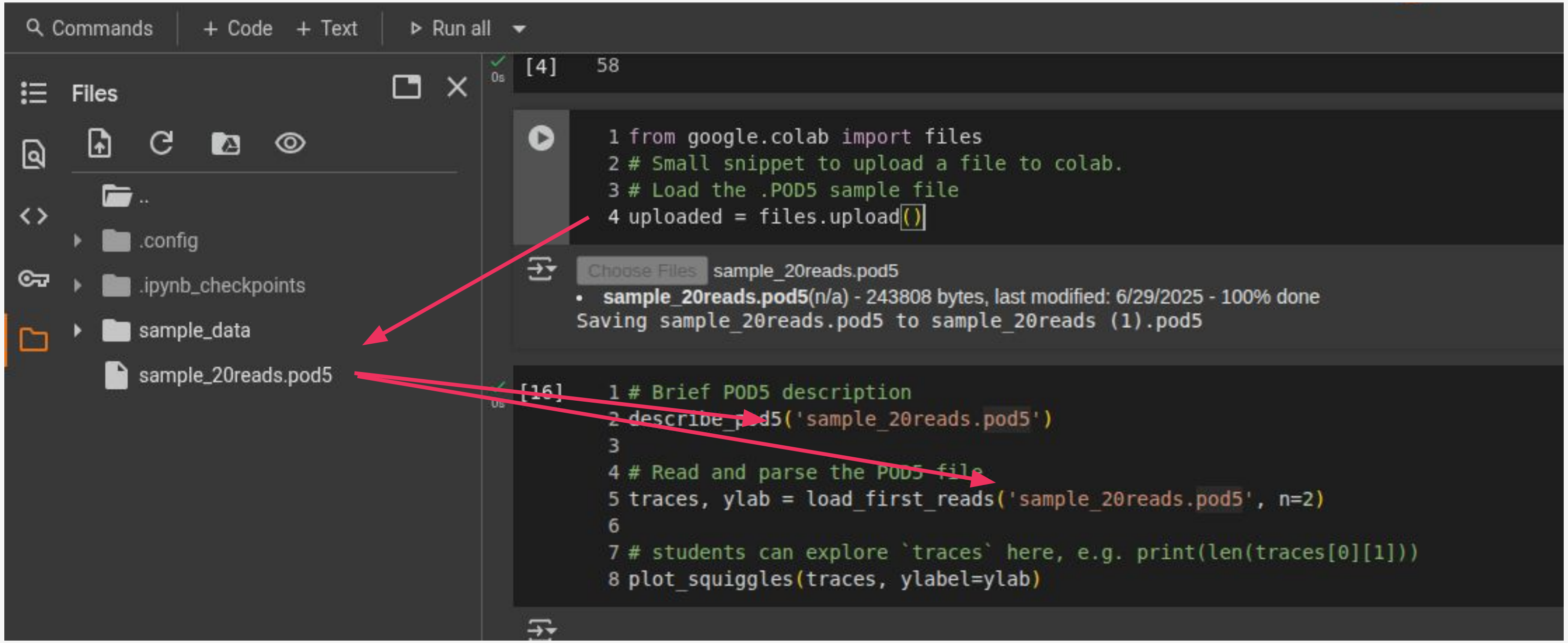
```
from google.colab import files
# Small snippet to upload a file to colab.
# Load the .POD5 sample file
uploaded = files.upload()
```

```
sample_20reads.pod5
• sample_20reads.pod5(n/a) - 243808 bytes, last modified: 6/29/2025 - 100% done
Saving sample_20reads.pod5 to sample_20reads (1).pod5
```

```
[16]
# Brief POD5 description
describe_pod5('sample_20reads.pod5')

# Read and parse the POD5 file
traces, ylab = load_first_reads('sample_20reads.pod5', n=2)

# students can explore `traces` here, e.g. print(len(traces[0][1]))
plot_squiggles(traces, ylabel=ylab)
```

Comments on the code

Uploading files to work with Google Colab:



```
[4]   58
```

```python
from google.colab import files
# Small snippet to upload a file to colab.
# Load the .POD5 sample file
uploaded = files.upload()
```

```
Choose Files sample_20reads.pod5
• sample_20reads.pod5(n/a) - 243808 bytes, last modified: 6/29/2025 - 100% done
Saving sample_20reads.pod5 to sample_20reads (1).pod5
```

```python
[16]
# Brief POD5 description
describe_pod5('sample_20reads.pod5')

# Read and parse the POD5 file
traces, ylab = load_first_reads('sample_20reads.pod5', n=2)

# students can explore `traces` here, e.g. print(len(traces[0][1]))
plot_squiggles(traces, ylabel=ylab)
```

SALMON data

- **Salmon** is a **fast**, **alignment-free** (or quasi-alignment) tool for **quantifying transcript abundances** from RNA-Seq data.
- Uses **k-mers and lightweight mapping** to estimate expression without full alignment.
- **Bias-aware**: models fragment GC bias, positional bias, and sequence-specific bias to improve accuracy.
- Outputs **Transcripts Per Million (`TPM`)** per **isoform**.

SALMON data

How TPM is calculated:

Let:

r_i = number of reads mapping to transcript i

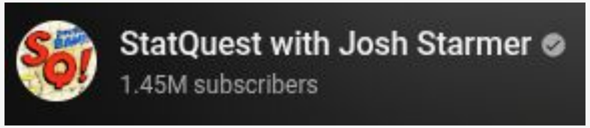
l_i = length of transcript i

1. **Calculate Reads per Kilobase (RPK):** $$RPK_i = \frac{r_i}{l_i/1000}$$

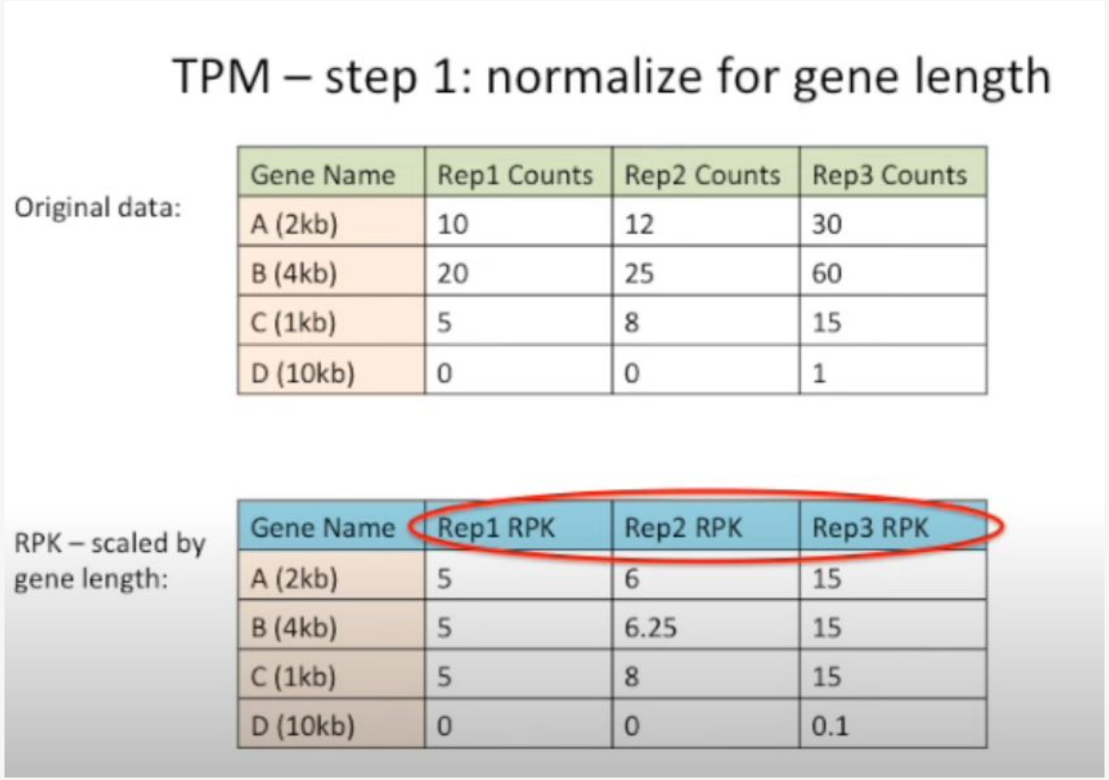
2. **Compute scaling factor (sum of RPKs):** $$\text{Scaling Factor} = \sum_j RPK_j$$

3. **Compute TPM:** $$TPM_i = \frac{RPK_i}{\text{Scaling Factor}} \times 10^6$$

https://www.youtube.com/watch?v=TTUrtCY2k-w



SALMON data



TPM – step 1: normalize for gene length

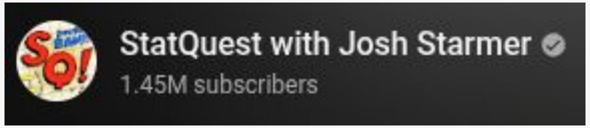
Original data:

Gene Name	Rep1 Counts	Rep2 Counts	Rep3 Counts
A (2kb)	10	12	30
B (4kb)	20	25	60
C (1kb)	5	8	15
D (10kb)	0	0	1

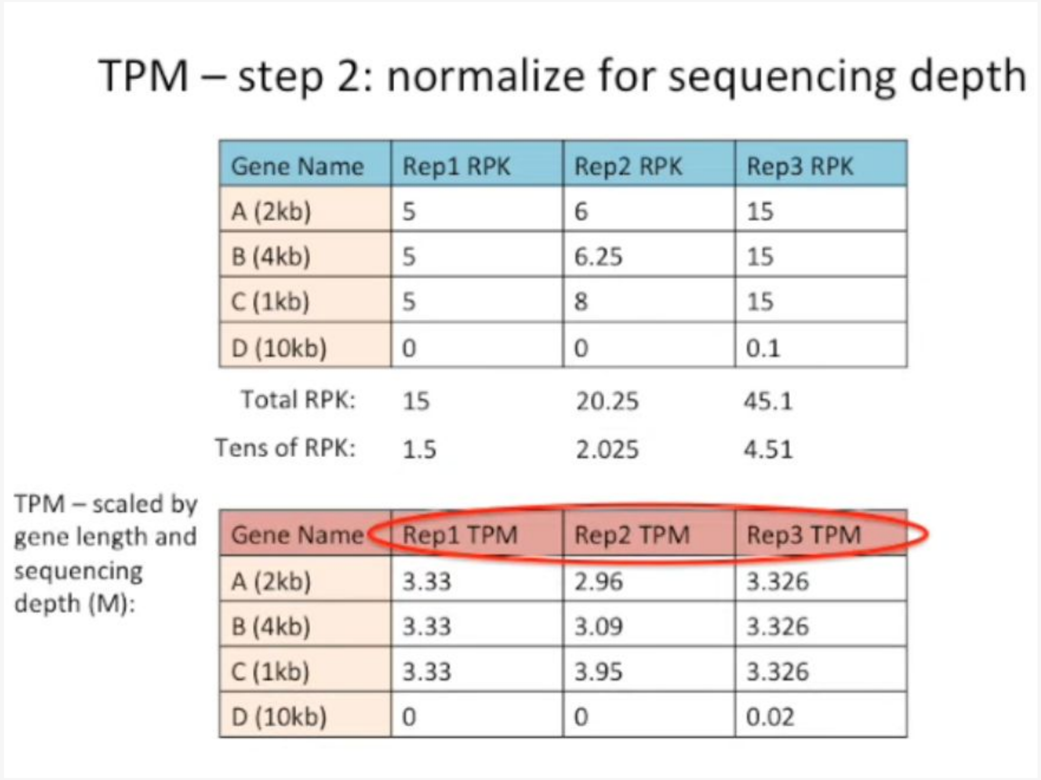
RPK – scaled by gene length:

Gene Name	Rep1 RPK	Rep2 RPK	Rep3 RPK
A (2kb)	5	6	15
B (4kb)	5	6.25	15
C (1kb)	5	8	15
D (10kb)	0	0	0.1

https://www.youtube.com/watch?v=TTUrtCY2k-w


StatQuest with Josh Starmer
1.45M subscribers

SALMON data


TPM – step 2: normalize for sequencing depth

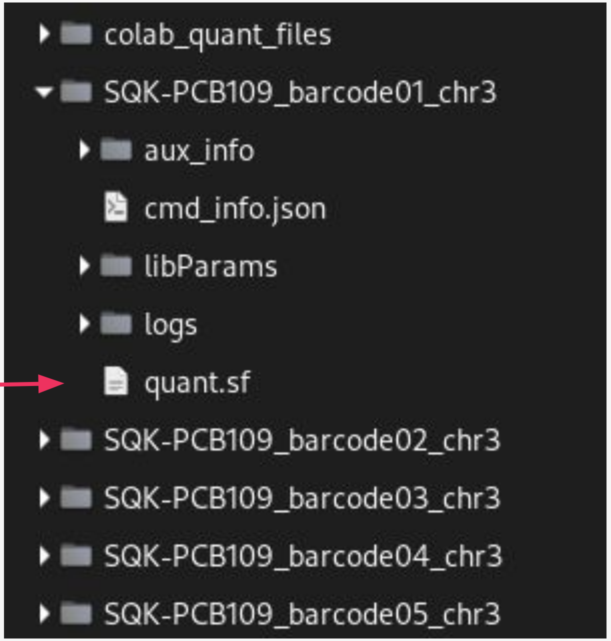
Gene Name	Rep1 RPK	Rep2 RPK	Rep3 RPK
A (2kb)	5	6	15
B (4kb)	5	6.25	15
C (1kb)	5	8	15
D (10kb)	0	0	0.1
Total RPK:	15	20.25	45.1
Tens of RPK:	1.5	2.025	4.51

TPM – scaled by gene length and sequencing depth (M):

Gene Name	Rep1 TPM	Rep2 TPM	Rep3 TPM
A (2kb)	3.33	2.96	3.326
B (4kb)	3.33	3.09	3.326
C (1kb)	3.33	3.95	3.326
D (10kb)	0	0	0.02

SALMON OUTPUT

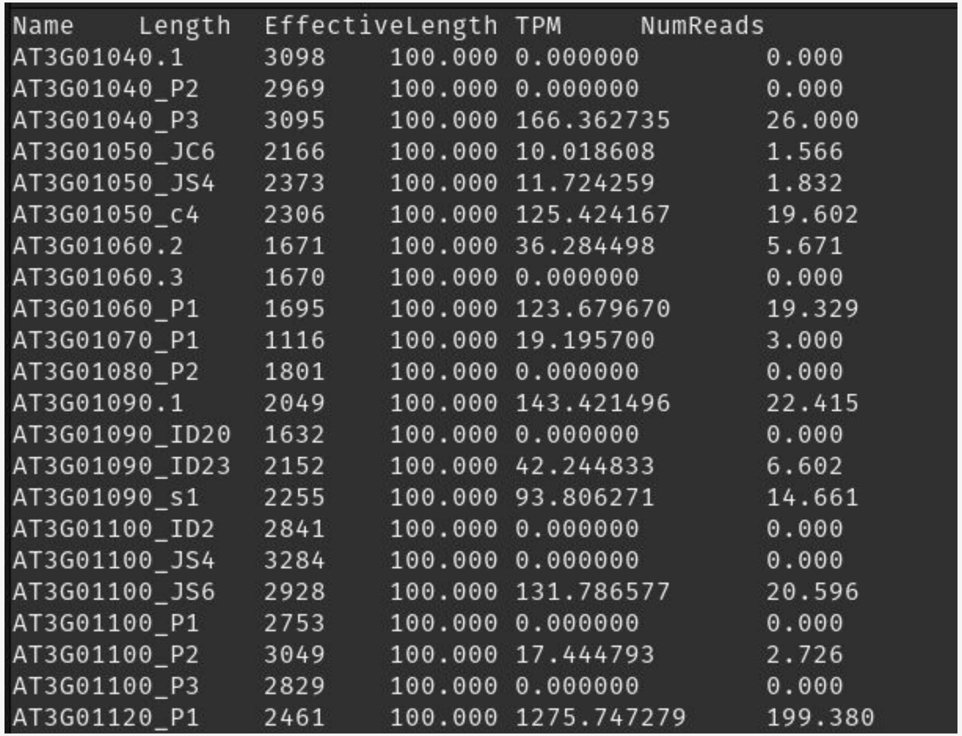
quant.sf



```
▸ colab_quant_files
▾ SQK-PCB109_barcode01_chr3
    ▸ aux_info
      cmd_info.json
    ▸ libParams
    ▸ logs
      quant.sf
▸ SQK-PCB109_barcode02_chr3
▸ SQK-PCB109_barcode03_chr3
▸ SQK-PCB109_barcode04_chr3
▸ SQK-PCB109_barcode05_chr3
```

SALMON OUTPUT

quant.sf



```
Name       Length   EffectiveLength TPM        NumReads
AT3G01040.1        3098     100.000 0.000000          0.000
AT3G01040_P2       2969     100.000 0.000000          0.000
AT3G01040_P3       3095     100.000 166.362735        26.000
AT3G01050_JC6      2166     100.000 10.018608         1.566
AT3G01050_JS4      2373     100.000 11.724259         1.832
AT3G01050_c4       2306     100.000 125.424167        19.602
AT3G01060.2        1671     100.000 36.284498         5.671
AT3G01060.3        1670     100.000 0.000000          0.000
AT3G01060_P1       1695     100.000 123.679670        19.329
AT3G01070_P1       1116     100.000 19.195700         3.000
AT3G01080_P2       1801     100.000 0.000000          0.000
AT3G01090.1        2049     100.000 143.421496        22.415
AT3G01090_ID20     1632     100.000 0.000000          0.000
AT3G01090_ID23     2152     100.000 42.244833         6.602
AT3G01090_s1       2255     100.000 93.806271         14.661
AT3G01100_ID2      2841     100.000 0.000000          0.000
AT3G01100_JS4      3284     100.000 0.000000          0.000
AT3G01100_JS6      2928     100.000 131.786577        20.596
AT3G01100_P1       2753     100.000 0.000000          0.000
AT3G01100_P2       3049     100.000 17.444793         2.726
AT3G01100_P3       2829     100.000 0.000000          0.000
AT3G01120_P1       2461     100.000 1275.747279       199.380
```

SALMON OUTPUT

Name:

Isoform name from our transcriptome annotation.

Length:

Length of the gene based on the reference.

EffectiveLength:

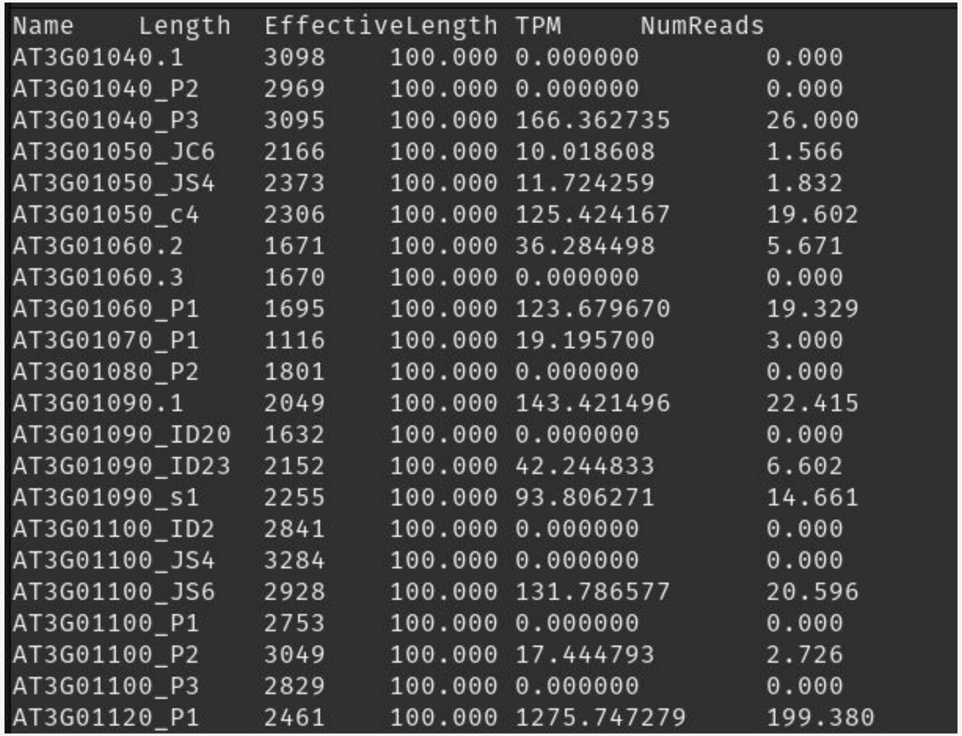
Important in Illumina data – represents how many reads could fit along a transcript, adjusted for fragment size and biases.

TPM:

Transcripts per million (normalized counts)

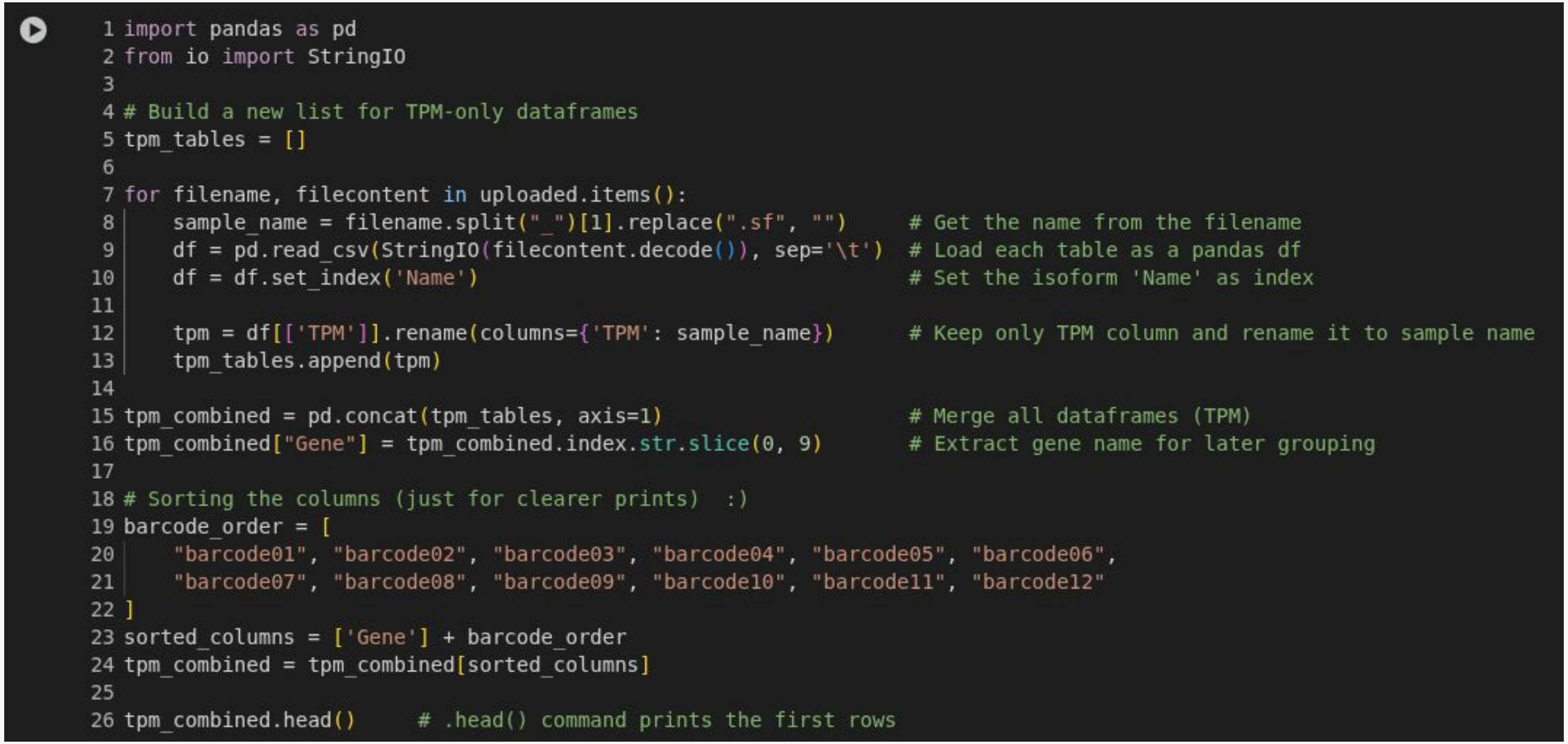
NumReads:

How many reads Salmon estimates came from this transcript.



```
Name        Length  EffectiveLength TPM       NumReads
AT3G01040.1       3098     100.000 0.000000          0.000
AT3G01040_P2      2969     100.000 0.000000          0.000
AT3G01040_P3      3095     100.000 166.362735        26.000
AT3G01050_JC6     2166     100.000 10.018608         1.566
AT3G01050_JS4     2373     100.000 11.724259         1.832
AT3G01050_c4      2306     100.000 125.424167        19.602
AT3G01060.2       1671     100.000 36.284498         5.671
AT3G01060.3       1670     100.000 0.000000          0.000
AT3G01060_P1      1695     100.000 123.679670        19.329
AT3G01070_P1      1116     100.000 19.195700         3.000
AT3G01080_P2      1801     100.000 0.000000          0.000
AT3G01090.1       2049     100.000 143.421496        22.415
AT3G01090_ID20    1632     100.000 0.000000          0.000
AT3G01090_ID23    2152     100.000 42.244833         6.602
AT3G01090_s1      2255     100.000 93.806271         14.661
AT3G01100_ID2     2841     100.000 0.000000          0.000
AT3G01100_JS4     3284     100.000 0.000000          0.000
AT3G01100_JS6     2928     100.000 131.786577        20.596
AT3G01100_P1      2753     100.000 0.000000          0.000
AT3G01100_P2      3049     100.000 17.444793         2.726
AT3G01100_P3      2829     100.000 0.000000          0.000
AT3G01120_P1      2461     100.000 1275.747279       199.380
```

Comments on the code



```python
import pandas as pd
from io import StringIO

# Build a new list for TPM-only dataframes
tpm_tables = []

for filename, filecontent in uploaded.items():
    sample_name = filename.split("_")[1].replace(".sf", "")     # Get the name from the filename
    df = pd.read_csv(StringIO(filecontent.decode()), sep='\t')  # Load each table as a pandas df
    df = df.set_index('Name')                                   # Set the isoform 'Name' as index

    tpm = df[['TPM']].rename(columns={'TPM': sample_name})      # Keep only TPM column and rename it to sample name
    tpm_tables.append(tpm)

tpm_combined = pd.concat(tpm_tables, axis=1)                    # Merge all dataframes (TPM)
tpm_combined["Gene"] = tpm_combined.index.str.slice(0, 9)       # Extract gene name for later grouping

# Sorting the columns (just for clearer prints)  :)
barcode_order = [
    "barcode01", "barcode02", "barcode03", "barcode04", "barcode05", "barcode06",
    "barcode07", "barcode08", "barcode09", "barcode10", "barcode11", "barcode12"
]
sorted_columns = ['Gene'] + barcode_order
tpm_combined = tpm_combined[sorted_columns]

tpm_combined.head()     # .head() command prints the first rows
```

Arabidopsis transcriptome notation

AT3G61860.1

AT3G61860.1

Chromosome number – Gene number

+ isoform suffix

SUPPA2

Isoform proportions

SUPPA2 Workflow – 3 Key Steps

1. 🧬 **Generate alternative splicing events** from a GTF annotation
   ➤ `suppa.py generateEvents -i annotation.gtf -o events -f ioi`

2. 📊 **Calculate PSI (Ψ) values** from transcript-level expression (e.g., TPMs)
   ➤ `suppa.py psiPerIsoform -i transcripts.ioi -e sample1.tpm,sample2.tpm -o psi_values.psi`

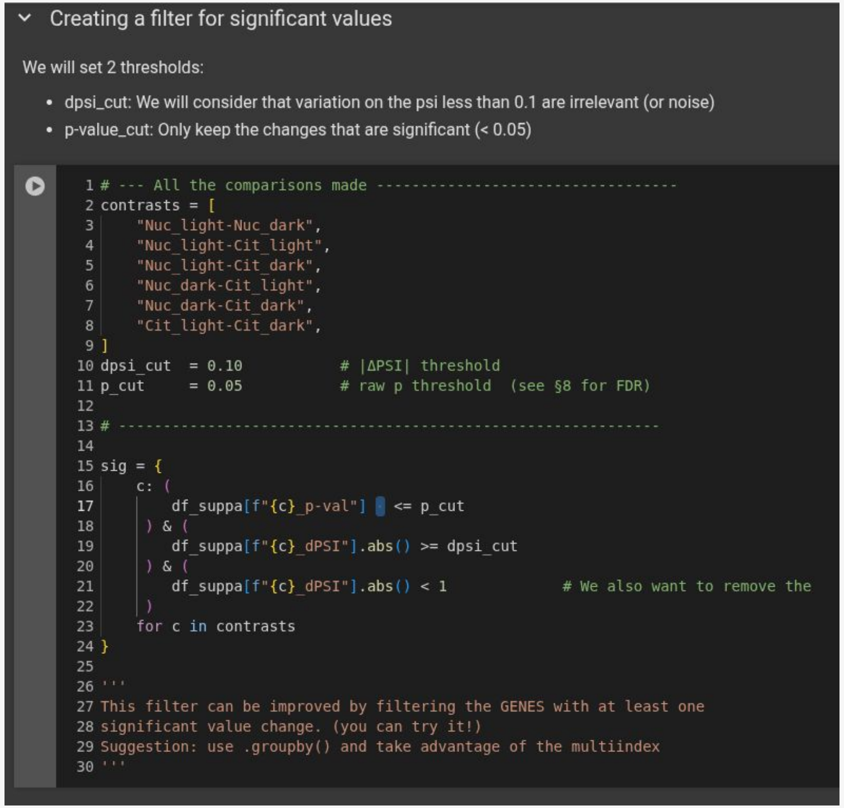
3. 📈 **Compare conditions to get ΔPSI and p-values**
   ➤
```
suppa.py diffSplice -m empirical -gc
         -i transcripts.ioi
         --psi condition1.psi condition2.psi
         --tpm condition1.tpm condition2.tpm
         -o diff_splicing
```

SUPPA 2 – In a nutshell

PSI (Ψ) and **delta PSI (ΔΨ)**

GENE	ISOFORM	TPM (cond 1)	PSI (cond 1)	TPM (cond 2)	PSI (cond 2)	Delta PSI
Gene_A	**Isoform_1**	10	0.1	50	0.625	+ 0.525
Gene_A	**Isoform_2**	70	0.7	30	0.375	- 0.325
Gene_A	**Isoform_3**	20	0.2	0	0	- 0.2

Setting some filters



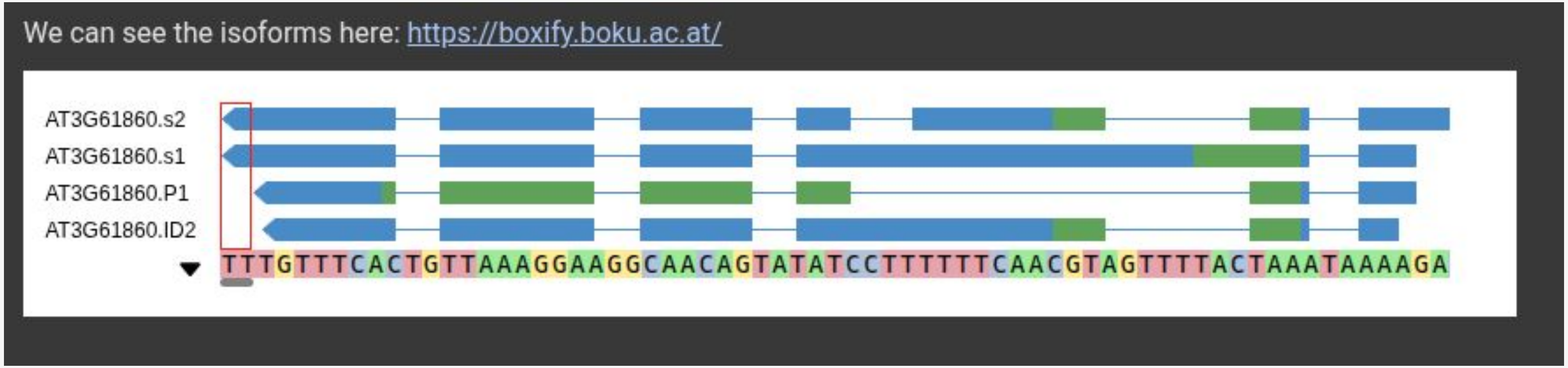
Creating a filter for significant values

We will set 2 thresholds:

- dpsi_cut: We will consider that variation on the psi less than 0.1 are irrelevant (or noise)
- p-value_cut: Only keep the changes that are significant (< 0.05)

```python
# --- All the comparisons made ----------------------------------
contrasts = [
    "Nuc_light-Nuc_dark",
    "Nuc_light-Cit_light",
    "Nuc_light-Cit_dark",
    "Nuc_dark-Cit_light",
    "Nuc_dark-Cit_dark",
    "Cit_light-Cit_dark",
]
dpsi_cut  = 0.10           # |ΔPSI| threshold
p_cut     = 0.05           # raw p threshold  (see §8 for FDR)

# -------------------------------------------------------------

sig = {
    c: (
        df_suppa[f"{c}_p-val"]  <= p_cut
    ) & (
        df_suppa[f"{c}_dPSI"].abs() >= dpsi_cut
    ) & (
        df_suppa[f"{c}_dPSI"].abs() < 1             # We also want to remove the
    )
    for c in contrasts
}

'''
This filter can be improved by filtering the GENES with at least one
significant value change. (you can try it!)
Suggestion: use .groupby() and take advantage of the multiindex
'''
```

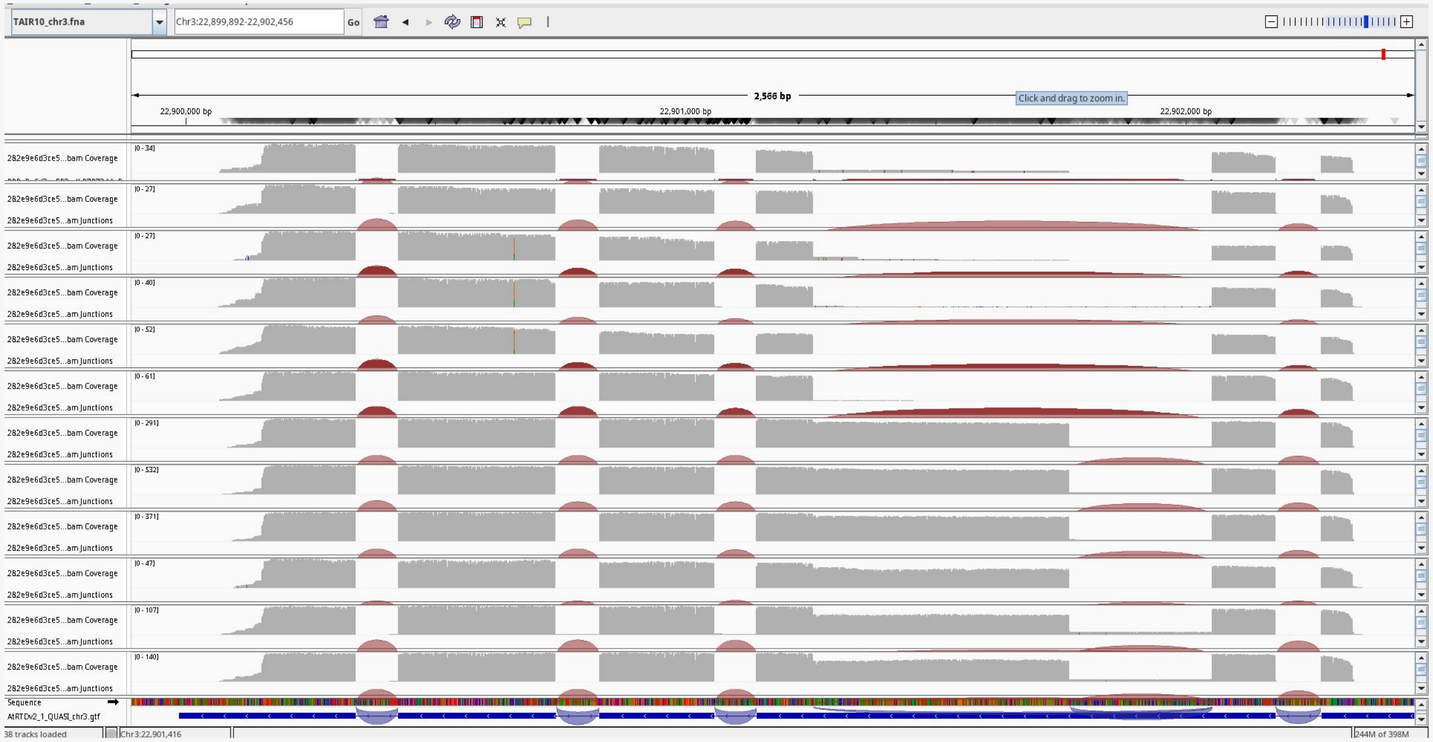
Visualizing isoforms


We can see the isoforms here: https://boxify.boku.ac.at/

AT3G61860.s2
AT3G61860.s1
AT3G61860.P1
AT3G61860.ID2
TTTGTTTCACTGTTAAAGGAAGGCAACAGTATATCCTTTTTTCAACGTAGTTTTACTAAATAAAAGA

https://boxify.boku.ac.at/

Integrative Genomics Viewer - IGV



https://igv.org/